/////////////////////////////////////////////////////////////////

// Final Project Milestone 1

// Aid Management Application

// Version 1.0

// Date Oct 20, 2017

// Author Eunhee Kim

// Description

// The Date class

// Class definaition and function declarations

//

//

// Revision History

// -----------------------------------------------------------

// Name               Date                 Reason

/////////////////////////////////////////////////////////////////

#ifndef SICT\_DATE\_H

#define SICT\_DATE\_H

#define NO\_ERROR   0

#define CIN\_FAILED 1

#define YEAR\_ERROR 2

#define MON\_ERROR  3

#define DAY\_ERROR  4

namespace sict{

//Pre\_defined constants

const int min\_year = 2000;

const int max\_year = 2030;

        //Define Date class

        class Date{

                //Private members

                private:

                int Dyear;

                int Dmonth;

                int Dday;

                long comparator;

                int error;

                int mdays(int month, int year) const;

                void errCode(int errorCode);

                //Public members

                public:

                Date();

                Date(int year, int month, int day);

                bool operator==(const Date& rhs) const;

                bool operator!=(const Date& rhs) const;

                bool operator<(const Date& rhs) const;

                bool operator>(const Date& rhs) const;

                bool operator<=(const Date& rhs) const;

                bool operator>=(const Date& rhs) const;

                int errCode() const;

                bool bad() const;

                std::istream& read(std::istream& istr);

                std::ostream& write(std::ostream& ostr) const;

        };

        //Helper functions

        std::ostream& operator<<(std::ostream& ostr, const Date& rhs);

        std::istream& operator>>(std::istream& istr, Date& rhs);

}

#endif

/////////////////////////////////////////////////////////////////

// Final Project Milestone 1

// Aid Management Application

// Version 1.0

// Date Oct 20, 2017

// Author Eunhee Kim

// Description

// The Date class to be used to hold the expiry date of the

// perishable items, so this file contains function definition.

//

//

// Revision History

// -----------------------------------------------------------

// Name               Date                 Reason

/////////////////////////////////////////////////////////////////

#include <iostream>

#include <iomanip>

#include "Date.h"

using namespace std;

namespace sict {

        //Default constructor

        Date::Date(){

                Dyear = 0;

                Dmonth = 0;

                Dday = 0;

                comparator = 0;

                error = NO\_ERROR;

        }

        //In its parameters integer balues for the year, month and day check if each number is in range

        Date::Date(int year, int month, int day){

                if(year < min\_year || year > max\_year){

                        Dyear = 0;

                        Dmonth = 0;

                        Dday = 0;

                        comparator = 0;

                        errCode(YEAR\_ERROR);

                }else if(month < 1 || month > 12){

                        Dyear = 0;

                        Dmonth = 0;

                        Dday = 0;

                        comparator = 0;

                        errCode(MON\_ERROR);

                }else if(mdays(month, year) < day || day < 1){

                        Dyear = 0;

                        Dmonth = 0;

                        Dday = 0;

                        comparator = 0;

                        errCode(DAY\_ERROR);

                }else{

                        Dyear = year;

                        Dmonth = month;

                        Dday = day;

                        comparator = year \* 372 + month \* 13 + day;

                        errCode(NO\_ERROR);

                }

        }

        //This operator return the result whether two objects are same

        bool Date::operator==(const Date& rhs) const{

                if(bad() || rhs.bad() || comparator == 0 || rhs.comparator == 0){

                        return false;

                }else{

                        return comparator == rhs.comparator;

                }

        }

        //This operator return the result whether two objects are not same

        bool Date::operator!=(const Date& rhs) const{

                if(bad() || rhs.bad() || comparator == 0 || rhs.comparator == 0){

                        return false;

                }else{

                        return comparator != rhs.comparator;

                }

        }

        //This operator returns the result whether current object is less than another Date object

        bool Date::operator<(const Date& rhs) const{

                if(bad() || rhs.bad() || comparator == 0 || rhs.comparator == 0){

                        return false;

                }else{

                        return comparator < rhs.comparator;

                }

        }

        //This operator returns the result whether current object is greater than another Date object

        bool Date::operator>(const Date& rhs) const{

                if(bad() || rhs.bad() || comparator == 0 || rhs.comparator == 0){

                        return false;

                }else{

                        return comparator > rhs.comparator;

                }

        }

        //This operator returns the result whether current object is less than or equals to another Date object

        bool Date::operator<=(const Date& rhs) const{

                if(bad() || rhs.bad() || comparator == 0 || rhs.comparator == 0){

                        return false;

                }else{

                        return comparator <= rhs.comparator;

                }

        }

        //This operator returns the result whether the current object is greater than or equals to other Date object

        bool Date::operator>=(const Date& rhs) const{

                if(bad() || rhs.bad() || comparator == 0 || rhs.comparator == 0){

                        return false;

                }else{

                        return comparator >= rhs.comparator;

                }

        }

        //Retruns the error state as an error code value

        int Date::errCode() const{

                return error;

        }

        //Returns true if the error state is not NO\_ERROR

        bool Date::bad() const{

                return error != NO\_ERROR;

        }

       //Reads the date from the console that an std::istream object

        std::istream& Date::read(std::istream& istr){

                char c1, c2;

                istr >> Dyear >> c1 >> Dmonth >> c2 >> Dday;

                if(istr.fail() == true){

                        errCode(CIN\_FAILED);

                }

                else{

                        if(Dyear < min\_year || Dyear > max\_year){

                                Dyear = 0;

                                Dmonth = 0;

                                Dday = 0;

                                comparator = 0;

                                errCode(YEAR\_ERROR);

                        }else if(Dmonth < 1 || Dmonth > 12){

                                Dyear = 0;

                                Dmonth = 0;

                                Dday = 0;

                                comparator = 0;

                                errCode(MON\_ERROR);

                        }else if(mdays(Dmonth, Dyear) < Dday || Dday < 1){

                                Dyear = 0;

                                Dmonth = 0;

                                Dday = 0;

                                comparator = 0;

                                errCode(DAY\_ERROR);

                        }else{

                                comparator = Dyear \* 372 + Dmonth \* 13 + Dday;

                                errCode(NO\_ERROR);

                        }

                 }

                return istr;

        }

        //Writes the date to an std::ostream object and then returns a reference to the std::ostream object

        std::ostream& Date::write(std::ostream& ostr) const{

                ostr << Dyear << "/" << setw(2) << setfill('0') << Dmonth << "/" << setw(2) << setfill('0') << Dday;

                return ostr;

        }

        //std::ostream object to print a date to the console

        std::ostream& operator<<(std::ostream& ostr, const Date& rhs){

                rhs.write(ostr);

                return ostr;

        }

        //std::istream object to read a date from the console

        std::istream& operator>>(std::istream& istr, Date& rhs){

                rhs.read(istr);

                return istr;

        }

        //Sets the error state variable

        void Date::errCode(int errorCode){

                error = errorCode;

        }

        // number of days in month mon\_ and year year\_

        //

        int Date::mdays(int mon, int year)const {

        int days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, -1 };

        int month = mon >= 1 && mon <= 12 ? mon : 13;

        month--;

        return days[month] + int((month == 1)\*((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0));

        }

}